Experiment-8

8. Implement a C program to perform symbol table operations.

Program:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_SIZE 100

struct SymbolTable {
   char label[10];
   char symbol[10];
   int addr;
};

struct SymbolTable symbolTable[MAX_SIZE];
int size = 0;

void insert();
void display();
int search(char lab[]);
void modify();
void deleteentry();

int main() {
   int choice;

   do {
      printf("\nSymbol Table Operations\n");
      printf("1. Insert\n");
      printf("2. Display\n");
      printf("3. Search\n");
      printf("4. Modify\n");
      printf("5. Delete\n");
      printf("6. Exit\n");
      printf("Enter your choice: ");
      scanf("%d", &choice);

      switch (choice) {
         case 1:
            insert();
            break;
         case 2:
            display();
            break;
         case 3: {
            char label[10];
            printf("Enter the label to search: ");
```

```
            scanf("%s", label);
            int index = search(label);
            if (index != -1)
                printf("Label found at index %d\n", index);
            else
                printf("Label not found\n");
            break;
        }
        case 4:
            modify();
            break;
        case 5:
            deleteentry();
            break;
        case 6:
            printf("Exiting program.\n");
            break;
        default:
            printf("Invalid choice. Please enter a number between 1 and 6.\n");
    }
} while (choice != 6);

return 0;
}

void insert() {
    if (size == MAX_SIZE) {
        printf("Symbol table is full. Cannot insert.\n");
        return;
    }

    printf("Enter the label: ");
    scanf("%s", symbolTable[size].label);

    // Check if the label already exists
    if (search(symbolTable[size].label) != -1) {
        printf("Label already exists. Duplicate cannot be inserted.\n");
        return;
    }

    printf("Enter the symbol: ");
    scanf("%s", symbolTable[size].symbol);
    printf("Enter the address: ");
    scanf("%d", &symbolTable[size].addr);

    size++;
    printf("Label inserted successfully.\n");
}

void display() {
```

```
    if (size == 0) {
        printf("Symbol table is empty.\n");
        return;
    }

    printf("\nSymbol Table\n");
    printf("Index\tLabel\tSymbol\tAddress\n");
    for (int i = 0; i < size; i++) {
        printf("%d\t%s\t%s\t%d\n", i, symbolTable[i].label, symbolTable[i].symbol,
symbolTable[i].addr);
    }
}

int search(char lab[]) {
    for (int i = 0; i < size; i++) {
        if (strcmp(symbolTable[i].label, lab) == 0) {
            return i;  // Return index if found
        }
    }
    return -1;  // Return -1 if not found
}

void modify() {
    char label[10];
    printf("Enter the label to modify: ");
    scanf("%s", label);

    int index = search(label);
    if (index == -1) {
        printf("Label not found. Cannot modify.\n");
        return;
    }

    printf("Enter the new symbol: ");
    scanf("%s", symbolTable[index].symbol);
    printf("Enter the new address: ");
    scanf("%d", &symbolTable[index].addr);

    printf("Modification successful.\n");
}

void deleteentry() {
    char label[10];
    printf("Enter the label to delete: ");
    scanf("%s", label);

    int index = search(label);
    if (index == -1) {
        printf("Label not found. Cannot delete.\n");
        return;
```

```
    }

    // Move elements to fill the gap
    for (int i = index; i < size - 1; i++) {
        symbolTable[i] = symbolTable[i + 1];
    }

    size--;
    printf("Deletion successful.\n");
}
```

Output: